

# A Note on Almost Perfect Probabilistically Checkable Proofs of Proximity

Shlomo Jozeph

November 18, 2015

## Abstract

Probabilistically checkable proofs of proximity (PCPP) are proof systems where the verifier is given a 3SAT formula, but has only oracle access to an assignment and a proof. The verifier accepts a satisfying assignment with a valid proof, and rejects (with high enough probability) an assignment that is far from all satisfying assignments (for any given proof).

In this work, we focus on the type of computation the verifier is allowed to make. Assuming  $P \neq NP$ , there can be no PCPP when the verifier is only allowed to answer according to constraints from a set that forms a CSP that is solvable in  $P$ . Therefore, the notion of PCPP is relaxed to almost perfect probabilistically checkable proofs of proximity (APPCPP), where the verifier is allowed to reject a satisfying assignment with a valid proof, with arbitrary small probability.

We show, unconditionally, a dichotomy of sets of allowable computations: sets that have APPCPPs (which actually follows because they have PCPPs) and sets that do not. This dichotomy turns out to be the same as that of the Dichotomy Theorem, which can be thought of as dividing sets of allowable verifier computations into sets that give rise to NP-hard CSPs, and sets that give rise to CSPs that are solvable in  $P$ .

## 1 Introduction

PCP of proximity [1, 6] is a proof system where the verifier is given a 3SAT formula (or, equivalently, a circuit), and oracle access to an assignment and a proof. The verifier is expected to distinguish between two cases:

- (Completeness) The assignment satisfies the formula (we may also require that the proof is valid), for which the verifier must always accept.
- (Soundness) The assignment is far from all satisfying assignments, for which the verifier must reject with a constant probability (e.g.  $1/2$ ), regardless of the given proof.

Usually, there are more limitations on the verifier, such as the amount of randomness it is allowed to use (e.g. logarithmic, polylogarithmic), and the number of location the verifier may query (e.g. constant, logarithmic).

In this work, we will not limit the verifier in these ways, but instead we limit the type of calculation the verifier is allowed to make. We fix a set of allowable computations of the verifier, who can only choose

a computation from the set. For example, if the set contains a single function, ONE\_IN\_THREE (which outputs 1 if exactly one of its three inputs are 1, and 0 otherwise) the verifier may query any three locations from the assignment and the proof (based on random coins and any computation), but the verifier accepts iff exactly one of the locations is set to true. It can be shown that such a PCPP exists.

In another example, which we will call the  $k$ -linear PCPP, the set contains all affine functions (modulo 2) over  $k$  bits. In this case, the verifier chooses a bit  $b \in \{0, 1\}$  and up to  $k$  locations to query. These choices can be made in any way. However, the verifier must accept iff the sum of the values in the chosen locations is equal to  $b$  (modulo 2).

A  $k$ -linear PCPP whose verifier uses a logarithmic amount of randomness and polynomial time cannot exist, under the assumption of  $P \neq NP$ : Suppose otherwise. We may treat each query of the verifier as a linear equation over the locations queried (the XOR of the locations queried should be equal to  $b$ , as chosen by the verifier). Given a circuit, we write all the linear equations corresponding to the verifier's choices. Since the verifier uses a logarithmic amount of randomness, there are polynomially many equations. If the circuit is satisfiable, there is a way to completely satisfy the verifier, which means that there is a solution for all equations. Otherwise, all assignments are far from satisfying, so there is no way to solve all equations. If a solution exists, it can be found via Gaussian elimination in polynomial time. Therefore, there is an algorithm to solve the circuit satisfaction problem in polynomial time.<sup>1</sup>

Hence, if we wish to consider arbitrary limitations of the power of the verifier, we should weaken the requirements of PCPPs. Taking inspiration from Håstad [8], who showed that it is hard to decide if a 3LIN instance is at least  $1 - \epsilon$  satisfiable or at most  $1/2 + \epsilon$  satisfiable, we weaken the completeness requirement to almost perfect completeness. That is, for every  $\epsilon > 0$ , we require a verifier (that may depend on  $\epsilon$ ) that has complete access to a 3SAT formula and oracle access to an assignment and a proof, to decide between the two cases:

- The assignment satisfies the formula (and the proof is valid), for which the verifier accepts with probability at least  $1 - \epsilon$ .
- The assignment is far from all satisfying assignments, for which the verifier accepts with probability at most  $1/2$  (we even allow at most  $1 - \omega(\epsilon)$ ).

Our result shows that, unconditionally, some sets of allowable computations do not have Almost Perfect Probabilistically Checkable Proofs of Proximity. These sets are exactly the ones that give rise to constraint satisfaction problems that are in  $P$ , under the assumption  $P \neq NP$ . On the other hand, any set that gives rise to CSP that is  $NP$ -complete has PCPP, which follows almost immediately from the existence of PCP of proximity [1, 6] and gadget reductions from 3SAT [9] (these reductions are stated more explicitly in [5]).

## 1.1 Definitions

The definitions related to constraints are based on the definitions from [5].

**Definition 1.1.** A *constraint* is a function  $f : \{0, 1\}^k \rightarrow \{0, 1\}$ .  $k$  is the *arity* of the constraint. The constraint is *satisfied* by an assignment  $x$  if  $f(x) = 1$ .

---

<sup>1</sup>Under the Exponential Time Hypothesis, a similar proof shows that even  $n^{1-\delta}$  randomness is not enough.

We define the constraints ID and NOT which have arity 1 and return the input and its negation, respectively.

**Definition 1.2.** A *constraint set* is a non-empty set of constraints.

For example, the constraint set corresponding to 2LIN contains two constraints,  $f_1$  and  $f_2$ , with  $f_1(x, y) = x \oplus y$  and  $f_2(x, y) = x \oplus y \oplus 1$ .

By the definition used in this paper, a constraint satisfaction problem is specified by a constraint set. Another way to specify a constraint satisfaction problem is by using a single function and allowing the usage of all possible negations. For example, our definition may need a constraint set containing 8 constraints to define 3SAT. It can also be defined by a single function,  $f(x, y, z) = x \vee y \vee z$ , if negations are allowed implicitly.

**Definition 1.3.** A *constraint application* is an ordered set  $\langle f, i_1, \dots, i_k \rangle$ , where  $f$  is a constraint of arity  $k$ , and each  $i_j$  is a natural number indicating the name of the variable.

The same variable name may appear several times in a constraint application.

**Definition 1.4.** Given a constraint set  $S$ , a *formula over  $S$  with  $n$  variables* is a (multi)set  $P$  containing constraint applications. For each  $c \in P$ ,  $c = \langle f, i_1, \dots, i_k \rangle$ , where  $k$  is the arity of  $f \in S$ , and  $i \in [n]$ . Usually,  $S$  and  $n$  will be implied from the context.

An *assignment*  $x \in \{0, 1\}^n$  satisfies a constraint application  $\langle f, i_1, \dots, i_k \rangle$  if  $f(x_{i_1}, \dots, x_{i_k}) = 1$ . A formula is  $\alpha$ -satisfied by an assignment  $x$  if an  $\alpha$ -fraction of the constraints in the formula are satisfied. A formula is  $\alpha$ -satisfiable if there is an assignment that  $\alpha$ -satisfies it. In the case of  $\alpha = 1$  we may omit  $\alpha$ .

A formula may contain several copies of the same constraint application. As a shorthand, we may say we give a constraint weight  $t$  when we mean that there are  $t$  copies of it in the formula. This can be generalized to non integer weights, since we will only care about fixed precision of weights (so we multiply all weights by a constant, and round to the nearest integer).

**Definition 1.5.** Given a constraint set  $S$  and  $0 \leq \varsigma < \kappa \leq 1$ , an  $(S, \kappa, \varsigma)$ -*Constraint Satisfaction Problem* ( $(S, \kappa, \varsigma)$ -CSP) is the following problem: Given a formula over  $S$ , decide whether there is an assignment to the variables satisfying at least  $\kappa$ -fraction of the constraint applications, or any assignment satisfies at most  $\varsigma$ -fraction of the variables.

A constraint set  $S$  will be called *NP-hard* if, for some  $\varsigma$ , solving the  $(S, 1, \varsigma)$ -CSP is NP-hard. A constraint set  $S$  will be called *APX-hard* if solving the  $(S, \kappa, \varsigma)$ -CSP, for some constants  $0 < \varsigma < \kappa < 1$ , is NP-hard.

Note that Every NP-hard constraint set is an APX-hard constraint set but the other direction is not true.

**Definition 1.6.** Given a constraint set  $S$ , an  $S$ -*verifier*  $V$  is a function on a 3SAT formula  $\varphi$ , and a random string  $r$  (of length bounded by some function of  $\varphi$ ). The verifier outputs  $f^r \in S$  of arity  $k$  and indices  $i_1^r, \dots, i_k^r$ . For a string  $s$ , the *acceptance probability* of  $V(\varphi)$  (the probability is on the random strings  $r$ ) on  $s$  is the probability (over  $r$ ) that  $f^r(s_{i_1^r}, \dots, s_{i_k^r}) = 1$ .

Note that an  $S$ -verifier defines a formula over  $S$ , by considering each index the verifier outputs as a variable, and defining a constraint application for every random string  $r$  by the output of the verifier on  $r$ . Then, the acceptance probability of  $V(\varphi)$  on  $s$  is the fraction of constraints satisfied by  $s$ .

**Definition 1.7.** An  $(S, \delta, d)$ -APPCPP (*Almost Perfect Probabilistically Checkable Proofs of Proximity*) for  $d > 1$ ,  $\delta > 0$  is a set of  $S$ -verifiers. For every  $\epsilon > 0$  small enough (that is, for some  $\Lambda > 0$ , for every  $\Lambda > \epsilon > 0$ ) the set contains a verifier  $V_\epsilon$ . For every integer  $n > 0$ , and  $\varphi$ , a 3SAT formula  $\varphi$  with  $n$  variables, the following holds:

- (Completeness) If  $\bar{a} \in \{0, 1\}^n$  is a satisfying assignment to  $\varphi$ , then there is a  $k > 0$  and  $\bar{\pi} \in \{0, 1\}^p$  such that the acceptance probability of  $V_\epsilon(\varphi)$  on  $(\bar{a}, \bar{\pi})$  is at least  $1 - \epsilon$ .
- (Soundness) If  $\bar{a} \in \{0, 1\}^n$  is  $(1 - \delta)$ -far from a satisfying assignment to  $\varphi$ , then for any  $\bar{\pi} \in \{0, 1\}^p$ , the acceptance probability of  $V_\epsilon(\varphi)$  on  $(\bar{a}, \bar{\pi})$  is at most  $1 - d\epsilon$ .

$\bar{\pi}$  is called a proof for  $\bar{a}$ . It is accepted in the completeness case, and rejected in the soundness case.

The definition does not restrict the verifiers in any way. For example, the verifiers may use an exponential amount of randomness, may require  $p$  to be exponential, and they may even be uncomputable.

**Definition 1.8.** A constraint  $f$  is said to be

- *0-valid* if  $f(0, \dots, 0) = 1$ .
- *1-valid* if  $f(1, \dots, 1) = 1$ .
- *Weakly positive* if  $f$  is equivalent to a conjunction of CNF clauses with at most one negated variable in each clause.
- *Weakly negative* if  $f$  is equivalent to a conjunction of CNF clauses with at most one non-negated variable in each clause.
- *Linear* if  $f$  is equivalent to a conjunction of linear equations.
- *2CNF* if  $f$  is equivalent to a conjunction of 2CNF clauses.
- *C-closed* if  $f(x) = f(\bar{x})$  for all  $x$ , where  $\bar{x}$  is the complement of the assignment  $x$ .

We use the same terminology for constraint sets, if all of the contained constraints satisfy the respective condition.

## 1.2 Results

As a first step, we state that any non-C-closed constraint set,  $S$ , that is NP-hard under the assumption  $P \neq NP$ , has  $(S, \delta, d)$ -APPCPP for some  $\delta$  and any  $d$ . This follows from the existence of PCPPs which have perfect completeness and constant soundness.

**Proposition 1.9.** *For every constraint set  $S$  that is NP-hard and not C-closed, there is a PCPP with perfect completeness and constant soundness that only uses constraints from  $S$ .*

*For every constraint set  $S$  that is APX-hard and not C-closed, there is a PCPP without perfect completeness (but constant completeness and soundness) that only uses constraints from  $S$ .*

Now we state that for any non-C-closed constraint set,  $S$ , that is not NP-hard under the assumption  $P \neq NP$ , there is no  $(S, \delta, d)$ -APPCPP for any  $\delta$  (and some  $d$  depending on  $\delta$ ).

**Theorem 1.10.** *The set of non-C-closed constraint sets that do not have  $(S, \delta, d)$ -APPCPP with some  $\delta > 0$ ,  $d > (\lceil \delta^{-1} \rceil + 2)^2$  is the set of non-C-closed constraint sets that are NP-complete, under the assumption  $P \neq NP$ .*

Combining these two results together, we get an unconditional dichotomy of non-C-closed constraint sets, into constraint sets that have PCPPs, and constraint sets that do not even have APPCPPs. The results may be extended to C-closed constraint sets, either by modifying the constraint sets to be non-C-closed, or by modifying the definition of APPCPP (see remark 2.1).

This dichotomy may help to explain why PCPPs are harder to construct than PCPs. Specifically, if we assume that  $P = NP$ , we may build any PCP using any constraint set. However, even such an incredible assumption does not help to construct even APPCPPs for some constraint sets.

Additionally, this dichotomy shows that PCPPs are stronger than locally testable codes. Locally testable codes can be constructed from good codes and PCPPs (see [1]), but even the existence of a locally testable and decodable code, where the tests are linear (the Hadamard code) does not help to generate APPCPPs over linear constraint sets (or a  $k$ -linear PCPP).

### 1.3 Related Work

The Dichotomy Theorem [9] states that every Boolean constraint satisfaction problem (CSP) is either NP-complete or solvable in polynomial time. Naturally, such a dichotomy is only meaningful if  $P \neq NP$ . However, Schaefer's proof of the Dichotomy Theorem can be thought of as showing an unconditional result, which is a weaker notion of PCPs of proximity: Suppose we are given a 3SAT formula (or, equivalently, a circuit), and oracle access to an assignment and a proof. We wish to use a verifier that queries the assignment and the proof at a few places to differentiate between two cases:

- (Completeness) The assignment satisfies the formula (we may also require that the proof is valid), for which the verifier must always accept.
- (Soundness) The assignment does not satisfy the formula (for any given proof), for which the verifier must reject at least once.

Then, only NP-hard sets may have such a weak PCPP.

Creignou [4] showed that CSPs can be classified into three classes by approximation hardness: CSPs that are NP-hard, CSPs that are APX-hard, and CSPs that can be maximized in polynomial time. Bulatov proved the Dichotomy Theorem for CSPs over variables that have three possible values [2]. Whether such a dichotomy holds for other non-Boolean CSPs is open (see [3] for a survey).

Probabilistically checkable proofs of proximity were introduced by Ben-Sasson et al. [1] and as assignment testers by Dinur and Reingold [6] as a tool helping composition of PCPs. PCPs of proximity are required to accept any correct proof with probability 1, unlike APPCPPs. The rejection probability may be defined in various ways. The most general one is a function of the distance of the assignment from the closest satisfying assignment. A simpler definition requires only a constant rejection probability if an assignment is constantly far from satisfying assignments.<sup>2</sup>

Guruswami proved a theorem (Theorem 1 in [7]) similar to proposition 2.4, with looser parameters.

---

<sup>2</sup>The proofs in this paper can be slightly simplified, if we only ask for a constant rejection probability.

## 2 Proofs

*Proof sketch of proposition 1.9.* PCPPs with perfect completeness and constant soundness where the verifier,  $V$ , makes a constant number of queries exist (see, for example [1]). There are gadget reductions from 3SAT to each NP-hard constraint set  $S$  (which is not C-closed), as shown by [9].

Consider a query of the verifier,  $V$  that queries  $k$  location. WLOG, we may assume that the verifier queries makes  $< 2^k$   $k$ SAT calculations. Using a reduction from  $k$ SAT to 3SAT (e.g. adding  $k - 3$  variables) and the gadget reduction from 3SAT to  $S$ , we change the this query of  $V$  to use only constraints from  $S$ . This costs us a polynomial (in  $k$ ) amount of calculation to be done by the verifier per (which we ignore in this paper) and a constant (depending on  $k$ ) addition of variables to the proof, per each query.

Similarly, for an APX-hard constraint set  $S$ , a PCPP with perfect completeness and constant soundness can be transformed into a PCPP without perfect completeness (but constant completeness and soundness) over  $S$  using a gadget reduction from 3SAT to  $S$  which was shown to exist by [5].  $\square$

*Proof of Theorem 1.10.* By Schaefer's classification [9], all constraint sets that are not NP-hard, under the assumption  $P \neq NP$ , are constraint sets of the following six types: Linear, Weakly positive, Weakly Negative, 2CNF, 1-valid, and 0-valid.

Linear constraint sets do not have APPCPPs due to proposition 2.2. Weakly positive and negative constraint sets do not have APPCPPs due to proposition 2.3. 2CNF constraint sets do not have APPCPPs due to proposition 2.4. 1-valid constraint sets accepts the all one assignment, so there's no  $(C, \delta, d)$ -APPCPP for  $C$  that is 1-valid (since there are 3SAT formulas that only accept the all zero assignment). Similarly, there is no  $(C, \delta, d)$ -APPCPP for  $C$  that is 0-valid.  $\square$

Note that the classification of C-closed constraint sets into those the have APPCPPs and those who do not does not depend on whether  $P = NP$  or  $P \neq NP$ .

*Remark 2.1.* If a constraint set  $S$  is C-closed, by adding any constraint  $c$  that is not C-closed,  $S \cup \{c\}$  becomes non-C-closed. The constraints ID and NOT are linear, weakly positive, weakly negative and 2CNF. The constraint ID is 1-valid, the constraint NOT is 0-valid. Therefore, to handle the case of C-closed constraint sets, we may allow the addition of the constraint ID or the constraint NOT without affecting the NP-hardness of the constraint set, thus converting a C-closed constraint set into a non-C-closed constraint set.

Another method to deal with C-closed constraint set is to relax the notion of APPCPP. That is, to accept assignments that are satisfying or the bit-wise inversion of satisfying assignment, and reject assignments that are far from both satisfying assignments and bit-wise inversions of satisfying assignments. This requires a further relaxation of APPCPP to only reject assignments that are  $1/2 - \delta$  far, since if there is a satisfying assignment, every string is  $1/2$ -close to it or its bit-wise inversion.

The proofs can be adapted to this model by defining a 3SAT formula on twice the number of variables: on one half of the variables the constraints are the same and the second half of the variables need to be true (by using the clause  $x$  for every variable  $x$ ). Then, the verifier will accept an all true assignment with some proof (for the case of weakly negative and 0-valid, the second half need to be false).

## 2.1 Linear Constraint Sets

**Proposition 2.2.** *There are no  $(C, \delta, \lceil \delta^{-1} \rceil + 2)$ -APPCPP, for  $C$  that is equivalent to a conjunction of linear equations.*

*Proof.* Let  $\varphi$  be a 2CNF formula on  $mn$  variables,  $\{x_i^j\}$ , where  $1 \leq i \leq n$  and  $1 \leq j \leq m$ , for an odd  $m$  such that  $\lceil \delta^{-1} \rceil \leq m \leq \lceil \delta^{-1} \rceil + 1$ . For every  $x_i^j$  and  $x_{i'}^j$ ,  $\varphi$  contains constraints requiring that  $x_i^j = x_{i'}^j$ ,  $(x_i^j \vee x_{i'}^j)$  and  $x_i^j \vee x_{i'}^j$ . For every  $j \neq j'$   $\varphi$  contains a constraint requiring that at most one of  $x_i^j$  and  $x_{i'}^{j'}$  is true  $(x_i^j \vee x_{i'}^{j'})$ . The only assignments completely satisfying  $\varphi$  are the assignments  $\{\alpha_j\}_{j=0}^m$ , where  $\alpha_j(x_i^j)$  is true and for any  $j' \neq j$ ,  $\alpha_j(x_{i'}^{j'})$  is false (the assignment  $\alpha_0$  is always false).

Suppose that there is a  $(C, \delta, \lceil \delta^{-1} \rceil + 2)$ -APPCPP for some constraint set  $C$  that is equivalent to a conjunction of linear equations. Then, there is some  $\Lambda > 0$ , such that for any  $\Lambda > \epsilon > 0$  there is a formula  $\psi_\epsilon$  (defined by the verifier  $V_\epsilon$ ) over  $C$  with the variables  $\{x_i^j\}$  and additional auxiliary variables, such that for every  $\alpha_j$  there is a proof  $\pi_j^\epsilon$  (on the auxiliary variables) such that  $(\alpha_j, \pi_j^\epsilon)$   $(1 - \epsilon)$ -satisfies  $\psi_\epsilon$ , and for any  $\alpha$  which is  $\delta$ -far from all  $\alpha_j$  and any  $\pi$ ,  $(\alpha, \pi)$  at most  $(1 - (\lceil \delta^{-1} \rceil + 2)\epsilon)$ -satisfies  $\psi_\epsilon$ . We show that for the assignment  $\beta$  that gives every variable from  $\{x_i^j\}$  true, and for some  $\pi$ , specifically for  $\pi = \oplus \pi_j^\epsilon$ , at least  $1 - (\lceil \delta^{-1} \rceil + 1)\epsilon$  of the constraints are true in  $\psi(\beta, \pi)$ . Note that  $\beta = \oplus \alpha_j$ .

The first step is removing all constraints from  $\psi_\epsilon$  that are not satisfied when assigning the variables  $(\alpha_j, \pi_j^\epsilon)$ , for all  $j > 0$ . This removes at most  $m\epsilon$  constraints from  $\psi$ . We show that all of the other constraints (an  $1 - m\epsilon$  fraction) are satisfied when using the assignment  $(\beta, \pi)$ . Consider a linear constraint  $\oplus y_k = b$ , where  $\{y_k\}$  are a set of variables and  $b \in \{0, 1\}$ . An assignment  $\alpha : \{y_k\} \rightarrow \{0, 1\}$  satisfying this constraint satisfies  $\oplus_k \alpha(y_k) = b$ . Given  $m$  assignments satisfying this constraint  $\gamma_j : \{y_k\} \rightarrow \{0, 1\}$ , the assignment  $\gamma = \oplus \gamma_j$  also satisfies the constraint:

$$\oplus_k \gamma(y_k) = \oplus_k (\oplus_j \gamma_j(y_k)) = \oplus_j (\oplus_k \gamma_j(y_k)) = \oplus_j b = b$$

since  $m$  is odd. A conjunction of linear constraints satisfied by all the  $\gamma_j$ 's is also satisfied by  $\gamma$ .

Therefore, the assignment  $(\beta, \pi) = (\oplus \alpha_j, \oplus \pi_j^\epsilon)$  satisfies all of the constraints in  $\psi$  not removed in the first step. However,  $\beta$  is  $1 - 1/m$  far from any of the  $\alpha_j$ 's, contradicting the definition of a  $(C, \delta, \lceil \delta^{-1} \rceil + 2)$ -APPCPP.  $\square$

## 2.2 Weakly Positive and Weakly Negative Constraint Sets

**Proposition 2.3.** *There are no  $(C, \delta, \lceil \delta^{-1} \rceil + 1)$ -APPCPP, for  $C$  that is equivalent to a conjunction of Horn clauses.*

*Proof.* Let  $\varphi$  be a 2CNF formula on  $mn$  variables,  $\{x_i^j\}$ , where  $1 \leq i \leq n$  and  $1 \leq j \leq m$ , for  $m = \lceil \delta^{-1} \rceil$ . For any  $x_i^j$ ,  $x_{i'}^j$ ,  $\varphi$  contains constraints requiring that  $x_i^j = x_{i'}^j$ ,  $(x_i^j \vee x_{i'}^j)$  and  $x_i^j \vee x_{i'}^j$ . For every  $j \neq j'$   $\varphi$  contains a constraint requiring that at most one of  $x_i^j$  and  $x_{i'}^{j'}$  is true  $(x_i^j \vee x_{i'}^{j'})$ . The only assignments completely satisfying  $\varphi$  are the assignments  $\{\alpha_j\}_{j=0}^m$ , where  $\alpha_j(x_i^j)$  is true and for any  $j' \neq j$   $\alpha_j(x_{i'}^{j'})$  is false (the assignment  $\alpha_0$  is always false).

Suppose that there is a  $(C, \delta, \lceil \delta^{-1} \rceil + 1)$ -APPCPP for some constraint set  $C$  that is equivalent to a conjunction of CNF clauses, where each clause has at most one negated variable. Then, there is some  $\Lambda > 0$ ,

such that for any  $\Lambda > \epsilon > 0$  there is a formula  $\psi_\epsilon$  (defined by the verifier  $V_\epsilon$ ) over  $C$  with the variables  $\{x_i^j\}$  and additional auxiliary variables, such that for every  $\alpha_j$  there is a proof  $\pi_j^\epsilon$  (on the auxiliary variables) such that  $(\alpha_j, \pi_j^\epsilon)$   $(1 - \epsilon)$ -satisfies  $\psi_\epsilon$  and for any  $\alpha$  which is  $\delta$ -far from all  $\alpha_j$  and any  $\pi$ ,  $(\alpha, \pi)$  at most  $(1 - (\lceil \delta^{-1} \rceil + 1)\epsilon)$ -satisfies  $\psi_\epsilon$ . We show that for the assignment  $\beta$  that gives every variable from  $\{x_i^j\}$  true, and for some  $\pi$ , specifically for  $\pi = \vee \pi_j^\epsilon$ , at least  $1 - \lceil \delta^{-1} \rceil \epsilon$  of the constraints are true in  $\psi(\beta, \pi)$ . Note that  $\beta = \vee \alpha_j$ .

The first step is removing all constraints from  $\psi_\epsilon$  that are not satisfied when assigning the variables  $(\alpha_j, \pi_j^\epsilon)$ , for all  $j > 0$ . This removes at most  $m\epsilon$  fraction of the constraints from  $\psi$ . We show that all of the other constraints (an  $1 - m\epsilon$  fraction) are satisfied when using the assignment  $(\beta, \pi)$ . Consider a clause  $\vee_{k>0} y_k \vee \overline{y_0}$ , where  $\{y_k\}$  are a set of variables. Given  $m$  assignments satisfying this clause  $\gamma_j : \{y_k\} \rightarrow \{0, 1\}$ , the assignment  $\gamma = \vee \gamma_k$  also satisfies the clause: If for all  $k$   $\gamma_k(y_0) = 0$ , then  $\gamma(y_0) = 0$ , so  $\gamma$  satisfies the clause. Otherwise, for some  $k$  there is an  $i$  such that  $\gamma_k(y_i) = 1$ , so  $\gamma(y_i) = 1$ , and  $\gamma$  satisfies the clause. The second argument also works for a clause of the form  $\vee_k y_k$ . A conjunction of Horn clauses satisfied by all the  $\gamma_j$ 's is also satisfied by  $\gamma$ .

Therefore, the assignment  $(\beta, \pi) = (\vee \alpha_j, \vee \pi_j^\epsilon)$  satisfies all of the constraints in  $\psi$  not removed in the first step. However,  $\beta$  is  $1 - 1/m$  far from any of the  $\alpha_j$ 's, contradicting the definition of a  $(C, \delta, \lceil \delta^{-1} \rceil + 1)$ -APPCPP.

Similarly, by negating all assignments used and all variables in the formulas, it can be shown that there is no  $(C, 1 - \delta, \lceil \delta^{-1} \rceil + 1)$ -APPCPP, when  $C$  is equivalent to a conjunction clauses which have at most one non-negated variable.  $\square$

## 2.3 2CNF Constraint Sets

**Proposition 2.4.** *There are no  $(C, \delta, (\lceil \delta^{-1} \rceil + 2)^2)$ -APPCPP, for  $C$  that is equivalent to a conjunction 2CNF clauses.*

*Proof.* Let  $\varphi$  be a 2CNF formula on  $mn$  variables,  $\{x_i^j\}$ , where  $1 \leq i \leq n$  and  $1 \leq j \leq m$ , for an odd  $m$  such that  $\delta^{-1} \leq m \leq \lceil \delta^{-1} \rceil + 1$ . For any  $x_i^j, x_{i'}^j, \varphi$  contains constraints requiring that  $x_i^j = x_{i'}^j$  ( $\overline{x_i^j} \vee x_{i'}^j$  and  $x_i^j \vee \overline{x_{i'}^j}$ ) and for every distinct  $j, j', j''$ ,  $\varphi$  contains a constraint requiring that at most two of  $x_i^j, x_{i'}^{j'}, x_{i''}^{j''}$  are true ( $\overline{x_i^j} \vee \overline{x_{i'}^{j'}} \vee \overline{x_{i''}^{j''}}$ ). The only assignments completely satisfying  $\varphi$  are the assignments  $\{\alpha_{j,k}\}$ , where  $\alpha_{j,k}(x_i^t)$  is true iff  $t \in \{j, k\}$  (and the assignment  $\alpha_0$  which gives 0 to all variables). Note that the assignment  $\alpha_{j,k}$  and  $\alpha_{k,,j}$  are the same, and we assume that  $\pi_{j,k}^\epsilon$  and  $\pi_{k,,j}^\epsilon$  are the same as well.

Suppose that there is a  $(C, \delta, (\lceil \delta^{-1} \rceil + 2)^2)$ -APPCPP for some constraint set  $C$  that is equivalent to a conjunction of 2CNF clauses. Then, there is some  $\Lambda > 0$ , such that for any  $\Lambda > \epsilon > 0$  there is a formula  $\psi_\epsilon$  (defined by the verifier  $V_\epsilon$ ) over  $C$  with the variables  $\{x_i^j\}$  and additional auxiliary variables, such that for every  $\alpha_{j,k}$  there is a proof  $\pi_{j,k}^\epsilon$  (on the auxiliary variables) such that  $1 - \epsilon$  of the constraints are true in  $\psi(\alpha_{j,k}, \pi_{j,k}^\epsilon)$ . We show that for the assignment  $\beta$  that assigns every variable from  $\{x_i^j\}$  true, and for some  $\pi$ , at least  $1 - (\lceil \delta^{-1} \rceil + 2)^2 \epsilon$  of the constraints in  $\psi_\epsilon$  are satisfied by  $(\beta, \pi)$ .

The first step is removing all constraints from  $\psi$  that are not satisfied when assigning the variables  $(\alpha_{j,k}, \pi_{j,k}^\epsilon)$  for all  $j, k$ . This removes at most  $m(m+1)\epsilon/2$  fraction of constraints from  $\psi$ . We show that all of the other constraints (an  $1 - m(m+1)\epsilon/2$  fraction) can be satisfied with  $(\beta, \pi)$ , for some  $\pi$  that will be constructed.



Any 2CNF formula can be represented by a graph on the literals with two directed edges  $(\bar{x}, y)$  and  $(\bar{y}, x)$  for every clause  $x \vee y$ , where each directed edge mean implication, that is, if the first literal is true, the second must be true as well (and if the first literal is false, the implication is true). Additionally, a variable and its negation must have opposite values.

Let  $\mathcal{M}_m$  be the set of all  $m \times m$  0/1 symmetric matrices. For  $A, B \in \mathcal{M}_m$ , we say that  $A \leq B$  iff  $A_{p,q} \leq B_{p,q}$  for all  $p, q$ .

Given a variable  $z$  of  $\psi$ , let  $V(z)$  be the (symmetric) matrix containing the  $m^2$  values  $z$  is assigned by  $\{(\alpha_{j,k}, \pi_{j,k})\}$ . For  $A \in \mathcal{M}_m$ , let  $V_A$  be the subset of variables of  $\psi$  such that  $V(z) = A$ . Let  $D^j$  be the matrix such that  $D^j_{p,q} = 1$  iff  $j \in \{p, q\}$ . Using this notation, we state the following facts:

- If  $A \neq B$ ,  $V_A \cap V_B = \emptyset$  (since a variable cannot be assigned two different values).
- For all  $i, j$ ,  $V(x_i^j) = D^j$ , so  $x_i^j \in V_{D^j}$  (by definition of  $\alpha_{j,k}$ ).
- If  $x \in V_A$  then  $\bar{x} \in V_B$ , for  $B$  such that  $A + B$  is the all ones matrix (since negations of literals must have opposite values).

Since we removed all constraints that are not satisfied in one of the  $(\alpha_{j,k}, \pi_{j,k})$ , if there is an index  $(q, p)$  such that  $A_{q,p} > B_{q,p}$  there cannot be an edge from  $V_A$  to  $V_B$  (otherwise,  $A_{q,p} = 1$ ,  $B_{q,p} = 0$ , so  $(\alpha_{q,p}, \pi_{q,p})$  assigns true to the variables in  $V_A$  but false to the variables in  $V_B$  which does not satisfy the implication constraint of the edge). Thus, there can only be edges from  $V_A$  to  $V_B$  if  $A \leq B$ .

Now we can construct  $\pi$ . For each  $A \in \mathcal{M}_m$ ,  $\pi$  assigns the same value to all variables in  $V_A$ . For all  $j$ ,  $\pi$  assigns 1 to the variables in  $V_{D^j}$ . To ensure that all on a directed path from  $V_{D^j}$  are satisfied, For all  $A$  and  $j$  such that  $D^j \leq A$ ,  $\pi$  assigns 1 to the variables in  $V_A$ . Since  $\pi$  must be a valid assignment,  $\pi$  assigns 0 to the variables in  $V_{1-D^j}$  and all variables in  $A$  such that  $A \leq 1 - D^j$ . Note that there are no  $A, j$  such that  $D^j \leq A \leq 1 - D^j$ , since then  $D^j \leq 1 - D^j$  and  $2D^j \leq 1$ , which is false ( $D^j$  has some elements of value 1). For all other variables,  $\pi$  gives the variables in  $V_A$  the majority of entries in  $A$ . By the construction of  $\pi$ , for all  $A \leq B$ , an edge from  $A$  to  $B$  must be satisfied by  $\pi$ , and for all  $A + B = 1$ , the variables in  $V_A$  have the opposite assignment to the variables in  $V_B$  (either by the majority condition, or by the fact that if  $D^j \leq A$  then  $B \leq 1 - D^j$ ). The variables of  $\beta$  are contained in  $\cup D_j$ , and are set to 1, so all the edges of the graph are satisfied (a  $1 - m(m+1)\epsilon/2$  fraction of the original constraints), but  $\beta$  is  $1 - 2/m$  far from any satisfying assignment to  $\psi$ .  $\square$

## Acknowledgments

Work supported in part by the Israel Science Foundation (grant No. 621/12).

## References

- [1] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 1–10. ACM, 2004.

- [2] Andrei A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 649–658, 2002.
- [3] AndreiA. Bulatov. On the CSP dichotomy conjecture. In *Computer Science Theory and Applications*, volume 6651 of *Lecture Notes in Computer Science*, pages 331–344. 2011.
- [4] Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995.
- [5] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*. Society for Industrial and Applied Mathematics, 2001.
- [6] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, December 2006.
- [7] Venkatesan Guruswami. On 2-query codeword testing with near-perfect completeness. In *ISAAC*, pages 267–276, 2006.
- [8] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001.
- [9] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, pages 216–226. ACM, 1978.